Logistics

• The final exam **time and location**
  – The final exam is from 9am to 12noon on Tuesday, January 20 in **McDonnell Hall A02**.
  – The exam will start and end promptly, so please do arrive on time.

• **Exam Format**
  – Closed book, closed note.
  – You may bring one 8.5-by-11 sheet (both sides) with notes in your own handwriting to the exam.
  – No electronic devices (e.g., calculators, laptops, and cell phones).
What to focus on

• focus on understanding basic issues, not memorizing details

• *For each algorithm*
  – understand how it works on typical input
  – Why do we care about this algorithm?
  – How is it different from other algorithms for the same problem?
  – When is it effective?
Material covered

• The exam will stress material covered since the midterm, including the following components.
  – Lectures 13–23.
  – Exercises 12–22.
  – Programming assignments 6–8
    • Wordnet, seamcarving, burrows-wheeler
Topics covered

Depth-first search
Kruskal's algorithm
Key-indexed counting
Knuth-Morris-Pratt substring search
RE to NFA
Run-length coding
Topics covered

Breadth-first search
Dijkstra's algorithm
LSD radix sort
Boyer-Moore substring search
R-way tries
Huffman coding
Topics covered

Topological sort
Bellman-Ford algorithm
MSD radix sort
Rabin-Karp substring search
Ternary search tries
LZW compression
Topics covered

Prim's algorithm
Ford-Fulkerson algorithm
3-way radix quicksort

Reductions
Burrows-Wheeler
Algorithm Analysis
1. Order of growth

public static int f3(int N) {
    if (N == 0) return 1;
    int x = 0;
    for (int i = 0; i < N; i++)
        x += f3(N-1);
    return x;
}

public static int f2(int N) {
    int x = 0;
    for (int i = 0; i < N; i++)
        for (int j = 0; j < i; j++)
            x++;
    return x;
}
2. Order of growth

Suppose that you collect the following memory usage data for a program as a function of the input size $N$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000</td>
<td>10,000 bytes</td>
</tr>
<tr>
<td>8,000</td>
<td>320,000 bytes</td>
</tr>
<tr>
<td>64,000</td>
<td>10,240,000 bytes</td>
</tr>
<tr>
<td>512,000</td>
<td>327,680,000 bytes</td>
</tr>
</tbody>
</table>

Estimate the memory usage of the program (in bytes) as a function of $N$ and use tilde notation to simplify your answer.
Graph Algorithms
3. Graph Search

- Run BFS, DFS (starting from vertex A)
- Identify one situation where you would need to use BFS instead of DFS.
- Identify one situation where you would need to use DFS instead of BFS.
4. Order traversals

List the vertices in preorder, post order and reverse post order (starting with node 0)
5. MST

- Run Kruskals

- Prims
6. MST Algorithm Design

Suppose you know the MST of a weighted graph $G$. Now, a new edge $v-w$ of weight $c$ is inserted into $G$ to form a weighted graph $G'$. Design an $O(V)$ time algorithm to determine if the MST in $G$ is also an MST in $G'$. You may assume all edge weights are distinct.

Your answer will be graded for correctness, clarity, and conciseness.

1. State the algorithm

2. Explain why your algorithm takes $O(V)$ time
7. Match Algorithms

- T9 texting in a cell phone
- 1D range search
- 2D range search
- Document similarity
- Traveling salesperson problem
- Web crawler
- Google maps
- PERT/CPM (Program Evaluation and Review Technique / Critical Path Method).

A. Trie
B. Hashing
C. 3-way radix quicksort
D. Binary search tree
E. Kd tree
F. Depth-first search
G. Breadth-first search
H. Dijkstra’s algorithm
I. Topological sort
J. Bellman-Ford
K. Enumerate permutations
8. Dijkstra’s algorithm
Strings
9. TST

1. List the words in alphabetical order (black nodes denote the end of a word)

2. Insert aaca to TST
10. String Sorting

Put an X in each box if the string sorting algorithm (the standard version considered in class) has the corresponding property.

<table>
<thead>
<tr>
<th></th>
<th>mergesort</th>
<th>LSD radix sort</th>
<th>MSD radix sort</th>
<th>3-way radix quicksort</th>
</tr>
</thead>
<tbody>
<tr>
<td>stable</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>in-place</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sublinear time (in best case)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fixed-length strings only</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
12. Regular Expression to NFA

Convert the RE $a^* \mid (b \mid c \mid d)^*$ into an equivalent NFA using the algorithm described in lecture, showing the result after applying each transformation.
13. KMP Table
Compression
14. LZW compression

You receive the following message encoded using LZW compression.

97
98
128
129
131
132
130

Decode the message (a -97, b-98 ... and next code starts - 128
15. MaxFlow-MinCut
17. Algorithm Design

In data compression, a set of binary code words is *prefix-free* if no code word is a prefix of another. For example, \{01, 10, 0010, 1111\} is prefix free, but \{01, 10, 0010, 10100\} is not because 10 is a prefix of 10100.

1. Design an efficient algorithm to determine if a set of binary code words is prefix-free

2. What is the order of growth of the worst-case running time of your algorithm as a function of \(N\) and \(W\), where \(N\) is the number of binary code words and \(W\) is the total number of bits in the input?

3. What is the order of growth of the memory usage of your algorithm?

What is the Burrows-Wheeler transform of

b a b a a b a c

What is the Burrows-Wheeler inverse transform of

7
b b b b a a a a a
21. counting memory

- standard data types
- object overhead – 16 bytes
- array overhead – 24 bytes
- references – 8 bytes
- Inner class reference – 8 bytes

```java
public class TwoThreeTree<Key extends Comparable<Key>, Value> {
    private Node root;

    private class Node {
        private int count; // subtree count
        private Key key1, key2; // the one or two keys
        private Value value1, value2; // the one or two values
        private Node left, middle, right; // the two or three subtrees
    }

    ...
}
```

- How much memory is needed for a 2-3 tree that holds N nodes?
22. String Sorting

<table>
<thead>
<tr>
<th>Original Input</th>
<th>LSD Radix Sort</th>
<th>Sorted</th>
<th>MSD Radix Sort</th>
<th>3-way String Quicksort (no shuffle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KISS ABBA ENYA ABBA ENYA ACDC SOAD SADE ABBA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENYA ACDC INXS ACDC ABBA ABBA WHAM CAKE ACDC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INXS AQUA DIDO AQUA AQUA AQUA ABBA CARS AQUA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STYX BECK CARS BECK ACDC BUSH MOBY JAYZ BECK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOAD BLUR ACDC BLUR SOAD BLUR BECK ABBA BLUR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACDC BUSH FUEL BUSH CAKE BECK ACDC ACDC BUSH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KORN CAKE BUSH CAKE MUSE CAKE SADE BECK CAKE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FUEL CARS ABBA CUDA CARS HOLE CARS DIDO WHAM CARS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUSH DIDO AQUA DIDO SADE DIDO FUEL DIDO DIDO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABBA ENYA CAKE ENYA BUSH ENYA CAKE KISS ENYA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WHAM FUEL BLUR FUEL RUSH FUEL HOLE BLUR FUEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAKE HOLE JAYZ HOLE BECK HOLE TSOL INXS HOLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLUR INXS BECK INXS FUEL INXS KORN ENYA INXS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUSE JAYZ HOLE JAYZ TSOL JAYZ CARS SOAD JAYZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BECK KISS KORN KISS WHAM KISS MUSE MOBY KISS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOBY KORN KISS KORN KORN KORN BUSH HOLE KORN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOLE MUSE TSOL TSOL DIDO MUSE RUSH KORN MOBY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSOL MOBY MOBY MOBY BLUR MOBY KISS AQUA MUSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JAYZ RUSH MUSE MUSE KISS RUSH AQUA TSOL RUSH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AQUA STYX SADE SADE INXS STYX BLUR STYX SADE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SADE SOAD WHAM WHAM CARS SOAD INXS FUEL SOAD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CARS SADE SOAD SOAD STYX SADE ENYA MUSE STYX</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIDO TSOL RUSH RUSH MOBY TSOL STYX BUSH TSOL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUSH WHAM STYX STYX JAYZ WHAM JAYZ RUSH WHAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0 1
23. Reductions

Consider the following two problems:

- **3SUM.** Given $N$ integers $x_1, x_2, \ldots, x_N$, are there three distinct indices $i, j, k$ such that $x_i + x_j + x_k = 0$?

- **3SUMPLUS.** Given $N$ integers $x_1, x_2, \ldots, x_N$ and an integer $b$, are there three distinct indices $i, j, k$ such that $x_i + x_j + x_k = b$?

(a) Show that 3SUM linear-time reduces to 3SUMPLUS. To demonstrate your reduction, give the 3SUMPLUS instance that you would construct to solve the following 3SUM instance: $x_1, x_2, \ldots, x_N$.

(b) Show that 3SUMPLUS linear-time reduces to 3SUM. To demonstrate your reduction, give the 3SUM instance that you would construct to solve the following 3SUMPLUS instance: $b, x_1, x_2, \ldots, x_N$. 